Exploration and Policy Reuse

Fernando Fernández **Manuela Veloso**

July 2005
CMU-CS-05-172

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was conducted while the first author was visiting Carnegie Mellon from the Universidad Carlos III de Madrid, supported by a generous grant from the Spanish Ministry of Education and Fullbright. The second author was partially sponsored by Rockwell Scientific Co., LLC under subcontract no. B4U528968 and prime contract no. W911W6-04-C-0058 with the US Army, and by BBNT Solutions, LLC under contract no. FA8760-04-C-0002 with the US Air Force. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the sponsoring institutions, the U.S. Government or any other entity.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **JUL 2005**	2. REPORT TYPE	3. DATES COVERED **00-00-2005 to 00-00-2005**
4. TITLE AND SUBTITLE **Exploration and Policy Reuse**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,School of Computer Science,Pittsburgh,PA,15213**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**16**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Keywords: Reinforcement Learning, Policy Reuse, Exploration Strategies.

Abstract

We define Policy Reuse as a learning technique guided by past policies offering the challenge of balancing among three choices: exploitation of the ongoing learned policy, exploration of random actions, and exploration towards the past policies. In this work we introduce a new exploration strategy, π-reuse, as an intelligent bias to reuse a past policy when learning a new one. Interestingly, this strategy also provides a similarity metric among a set of past policies and the new one. We therefore define a π-reuse based similarity metric between policies. We introduce a new algorithm that combines the selection and reuse of past policies using this similarity metric. We show empirical results that demonstrate the usefulness of our exploration strategy, π-reuse, as an intelligent bias to reuse past policies, and also, its effectiveness in defining similarity between policies.

1 Introduction

Policy Reuse can be defined as the capability of integrating past action policies in new learning processes. In this work, the motivation of Policy Reuse is to use the knowledge acquired to solve different tasks when learning a new one in the same domain. The domain defines how the agent behaves in the environment, i.e. the state transition function; each different task in the same domain is characterized through its reward function.

We introduce reusing of past policies in Reinforcement Learning as an exploration bias during a learning process. However, it is still a challenge, given that biasing the learning inherently complicates the exploration strategy. That is because in addition to the classical balance between exploring new states or exploiting the current policy, it adds a third factor of exploiting the past policy. However, this balance has been successfully found in other problems like path planning, where reusing waypoints used in past plans has demonstrated to be useful to solve new planning problems [3].

In this work we introduce a new exploration strategy, called π-reuse, that integrates a past policy in an ongoing learning process. This strategy assumes that a supervisor provides the action policy used to bias the exploration. We demonstrate that the learning performance of the new policy can be improved by biasing the exploratory process with the past policy, depending on whether the policy provided by the supervisor solved a task which was "similar" to the new one or not.

However, the application of Policy Reuse is much more complex if we receive a set of policies, because then we need to select the most accurate one to bias the learning of the new task. In this sense, we exploit the capabilities of Policy Reuse to define a similarity metric between the past policies and the new one. This similarity metric is based on the performance obtained when following the π-reuse strategy to solve the new task with the different past policies. The higher the performance is, the higher the similarity is.

The report is organized as follow. The next section summarizes related work, focusing on exploration strategies and in policy reuse methods. Section 3 formalizes the concepts of task and domain. Section 4 introduces the new exploration strategy, π-reuse. Section 5 describes the experiments performed, whose results motivate the definition of the similarity metric presented in Section 6. Section 7 discusses the main conclusions and further research.

2 Related Work

This work is motivated by two main research areas, the reuse of past policies and exploration strategies. Reusing sub-policies which were learned for a different but related task can be used to minimize the experience required to solve a new task. For instance a subproblem of an MDP can be defined as a new MDP where the state space is a subset of the original one. Then, the original MDP can be solved reusing policies learned for different subproblems [2]. Intra-Option Learning [9] and TTrees [13] also reuse macro-actions to learn new action policies, in both cases, in Semi-Markov Decision Processes. Hierarchical RL uses different abstraction levels to organize subtasks [5].

Some methods try to learn environment independent knowledge so the learned knowledge can be used for similar tasks in different scenarios [11]. Reusing the Q function that represents a policy learned for a task can be useful if it is similar to the new one [4]. However, it requires the Q function to be available, and not only the policy.

Balancing exploration and exploitation is typically exemplified with the multi-armed bandit problem [8], and tries to define whether to explore new or exploit the knowledge already acquired [1]. In the literature, different kinds of exploration strategies can be found. A random strategy always selects randomly the action to execute, without using the acquired knowledge. The ϵ-greedy strategy selects the best action suggested by the Q function with a probability of ϵ, and it selects a random action with probability of $(1-\epsilon)$. Boltzmann strategy ranks the actions, providing with a higher probability to the actions with a higher value of Q.

Directed exploration strategies memorize exploration-specific knowledge that is used for guiding the exploration search[10]. These strategies are based in heuristics that bias the learning so unexplored states tend to have a higher probability of being explored that recently visited ones. However, most of them require a model of the domain (the state transition function) to execute the heuristics.

Most of the previous examples are focused only on exploration or in reuse of sub-policies. Instead, our work focuses on policy as an exploration bias in the new learning problem, and we investigate such exploration strategies.

3 Domains and Tasks

Markov Decision Process [7] is represented with a tuple $<\mathcal{S},\mathcal{A},\delta,\mathcal{R}>$, where $\mathcal{S}$ is the set of all possible states, $\mathcal{A}$ is the set of all possible actions, δ is an unknown stochastic state transition function, $\delta:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\rightarrow\Re$, and $\mathcal{R}$ is an unknown stochastic reward function, $\mathcal{R}:\mathcal{S}\times\mathcal{A}\rightarrow\Re$. We focus in RL domains where different *tasks* can be solved. We introduce a task as a specific reward function, but the other concepts, $\mathcal{S}$, $\mathcal{A}$ and δ stay constant for all the tasks. Thus, we extend the concept of an MDP introducing two new concepts: domain and task. We characterize a domain, $\mathcal{D}$, as a tuple $<\mathcal{S},\mathcal{A},\delta>$. We define a task, Ω, as a tuple $<\mathcal{D},\mathcal{R}_\Omega>$, where $\mathcal{D}$ is a domain as defined before, and $\mathcal{R}_\Omega$ is the stochastic and unknown reward function.

In this work we assume that we are solving a task with absorbing goal states. Thus, if s_i is a goal state, $\delta(s_i,a,s_i)=1$, $\delta(s_i,a,s_j)=0$ for $s_i\neq s_j$, and $\mathcal{R}(s_i,a)=0$, for all $a\in\mathcal{A}$. A trial starts by locating the learning agent in a random position in the environment. Each trial finishes when a goal state is reached or when a maximum number of steps, say H, is achieved. Thus, the goal is to maximize the expected average reinforcement per trial, say W, as defined in equation 1:

$$W=\frac{1}{K}\sum_{k=0}^{K}\sum_{h=0}^{H}\gamma^h r_{k,h} \tag{1}$$

where γ ($0\leq\gamma\leq1$) reduces the importance of future rewards, and $r_{k,h}$ defines the immediate reward obtained in the step h of the trial k, in a total of K trials.

4 An Exploratory Strategy for Policy Reuse

The goal of this work is to describe how learning can be helped if different policies, which solve different tasks, are used in the learning of the action policy of another similar task. But first, we need to describe how only one past policy biases the learning of the new one.

4.1 Scope

We define an action policy, Π, as a function $\Pi : \mathcal{S} \rightarrow \mathcal{A}$. If the action policy was created to solve a defined task, Ω, the action policy is called Π_Ω. The gain, or average expected reward, received when executing an action policy Π in the task Ω is called W_Ω^Π. Lastly, an optimal action policy for solving the task Ω is called Π_Ω^*. Then, the scope of this section is the following:

- We need to solve the task Ω, i.e. learn Π_Ω^*.
- We have previously solved the set of tasks $\{\Omega_1, \ldots, \Omega_n\}$, so we have their respective optimal policies, $\{\Pi_{\Omega_1}^*, \ldots, \Pi_{\Omega_n}^*\}$
- Let's assume that there is a supervisor who, given Ω, tells us which is the most similar task, Ω_s to Ω. Thus, we know that the policy to reuse is Π_s^*.

Thus, in this section we assume that it exists a supervisor who provides a policy that solves a task similar to the one that we are trying to solve. A discussion on how similarities between tasks and their respective policies can be computed, and how to automatically estimate the policy to reuse, will be introduced in Section 6.

4.2 The π-reuse Exploration Strategy

We denote the old policy with Π^{old}, and the one we are currently learning with Π. We assume that we are using a direct RL method to learn the action policy, so we are learning its related Q function. Any algorithm can be used to learn the Q function, with the only requirement that it can learn off-policy, i.e. it can learn a policy while executing a different one, as Q-Learning does [14].

The goal of the π-reuse strategy is to balance random exploration, exploitation of the old policy, and exploitation of the new policy, which is being learned currently. The π-reuse strategy follows the past policy with a probability of ψ. However, with a probability of $1 - \psi$, it exploits the new policy. Obviously, random exploration is always required, so when exploiting the new policy, it follows an ϵ-greedy strategy, as is defined in Table 1. Lastly, the υ parameter allows to decay the value of ψ in each trial.

Thus, there are three probabilities involved: the probability of exploiting the past policy, the probability of using current policy, and the probability of acting randomly. These probabilities are shown in Figure 1, for input values of $H = 100$, $\psi = 1$ and $\upsilon = 0.95$. In this case the ϵ parameter is set in each step to $1 - \psi_h$.

The figure shows that in the initial steps of each trial, the past policy is exploited. As the number of steps increases, exploration also increases, while in the final steps of the trial, the new policy will be exploited. The transition from exploiting the past policy and exploiting the new one depends on the υ parameter. If this parameter is low, the transition occurs in the initial steps, while if it is high, the transition is delayed.

```
$\pi$-reuse ($\Pi_{old}, K, H, \psi, \upsilon$).
for $k = 1$ to $K$
    Set the initial state, $s$, randomly.
    Set $\psi_1 \leftarrow \psi$
    for $h = 1$ to $H$
        With a probability of $\psi_h$, $a = \Pi_{old}(s)$
        With a probability of $1 - \psi_h$, $a = \epsilon$-greedy$(\Pi_{new}(s))$
        Receive current state $s'$, and reward, $r_{k,h}$
        Update $Q^{\Pi_{new}}(s, a)$, and therefore, $\Pi_{new}$
        Set $\psi_{h+1} \leftarrow \psi_h \upsilon$
        Set $s \leftarrow s'$
$W = \frac{1}{K} \sum_{k=0}^{K} \sum_{h=0}^{H} \gamma^h r_{k,h}$
Return $W$ and $\Pi_{new}$
```

Table 1: π-reuse Exploration Strategy.

5 Experiments

In this section, we describe the experiments performed to demonstrate the usefulness of the exploration strategy defined above. But first, we describe the domain used.

5.1 Tasks in a Robot Navigation Domain

This domain consists of a robot moving inside of an office area, as shown in Figure 2, similar to the one used in other RL works [6, 12]. The environment is represented by walls, free positions and goal areas, all of them of size 1×1. The whole domain is $N \times M$ (24×21 in this case). The possible actions that the robot can execute are “North”, “East”, “South” and “West”, all of size one. The final position after each action is noised by a random variable following a uniform distribution in the range $(-0.20, 0.20)$. The robot knows its location in the space through continuous coordinates (x, y) provided by some localization system. In this work, we assume that we have the optimal uniform discretization of the state space (which consists of 24×21 regions). Furthermore, the robot has an obstacle avoidance system that blocks the execution of actions that would crash it into a wall. The goal in this domain is to reach the area marked with ’G’. When the robot reaches it, it is considered a successful trial, and it receives a reward of 1. Otherwise, it receives a reward of 0.

Figure 2 shows six different tasks in the same domain, Ω_1, Ω_2, Ω_3, Ω_4 and Ω, given that the goal states, and therefore, the reward functions, are different. Biasing the learning of Π (to solve Ω) using Π_1 (policy that solves Ω_1) seems to be useful given that both policies could be equal for a large number of states. However, what states share the same policy and what states do not is completely unknown a priori, given that both the reward function and the state transition function are unknown.

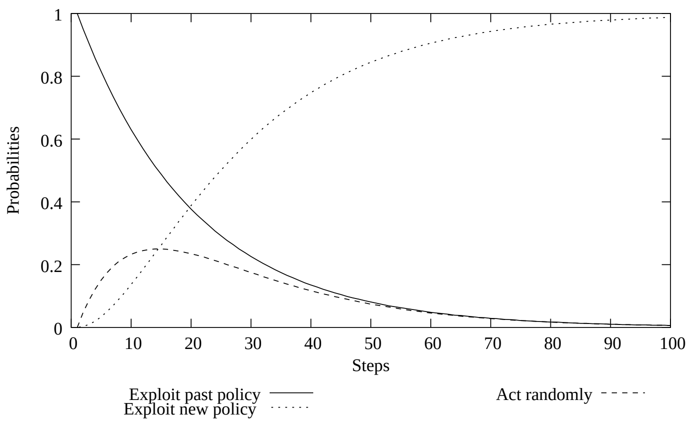


Figure 1: Evolution of the probabilities of exploring and exploiting in a trial for the π-reuse exploration strategy.

5.2 Description of the Learning Curves

In the following subsections, we will describe the experimental results of applying different exploration strategies for learning the task Ω, shown in Figure 2(e). For each of these strategies (and parameter settings), we will present two results showing two different curves, the learning curve, and the test curve.

The learning curve of each strategy describes the performance of such strategy in the learning process. Learning has been performed using the Q-Learning algorithm, for fixed parameters of $\gamma = 0.95$ and $\alpha = 0.05$, which empirically have demonstrated to be accurate for learning.

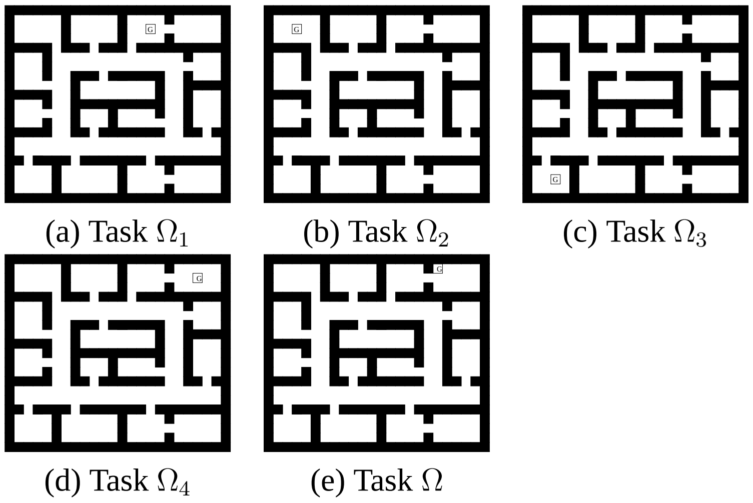
A learning trial consists of executing $K = 2000$ trials. Each trial consists on following the defined strategy until the goal is achieved or until the maximum number of steps, $H = 100$, is executed. In the figures containing the curves, the x axis shows the trial number. The y axis represents the gain obtained. Thus, a value of 0.2 for the trial 200 means that the average gain obtained in the 200 first trials has been 0.2.

The test curve represents the evolution of the performance of the policy while it is being learned. Each 100 trials of the learning process, the Q function learned up to that moment is stored. Thus, after the learning process, we can test all those policies. Each test consists on 1000 trials where the robot follows a completely greedy strategy. Thus, the x axis shows the learning trial in which that policy was generated, and the y axis show the result of the test, measured as the average number of steps executed to achieve the goal in the 1000 test trials.

For both the learning and test curves, the results provided are the average of ten executions. In the curves, error bars provide the standard deviation in the ten executions.

5.3 Learning from Scratch

We want to learn the task described in Figure 2(e). For comparison reasons, the learning and test processes have been executed firstly following different exploratory strategies that do not use any past policy. Specifically, we have used four different strategies. The first one is a random strategy.

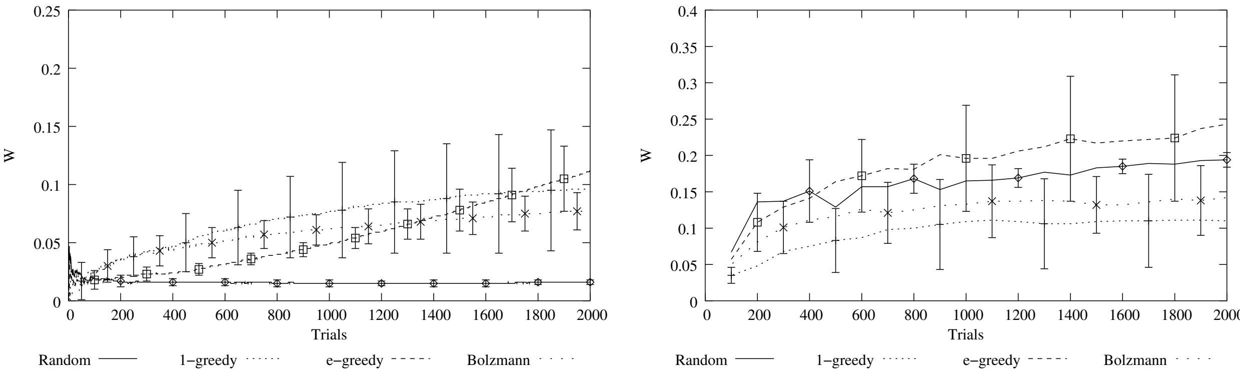


(a) Task Ω_1 (b) Task Ω_2 (c) Task Ω_3

(d) Task Ω_4 (e) Task Ω

Figure 2: Office Domain.

The second one is a completely greedy strategy. The third one is ϵ-greedy, for an initial value of $\epsilon = 0$, which is incremented by 0.0005 in each trial. Lastly, Boltzmann strategy has been used, initializing $\tau = 0$, and increasing it in 5 in each learning trial. Figure 3 shows the learning and test curves for all of them.



(a) Learning Curve (b) Test Curve

Figure 3: Learning and test evolution when learning from scratch

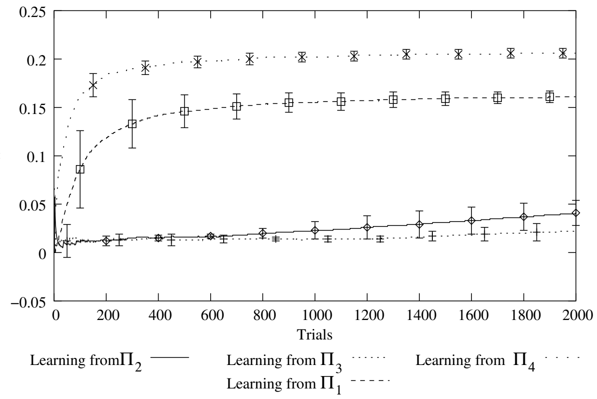
Figure 3(a) shows the learning curve. We see that when acting randomly, the average gain in learning is almost 0, given that acting randomly is a very poor strategy. However, when a greedy behavior is introduced, (strategy 1-greedy), the curve shows a slow increment, achieving values of almost 0.1. The problem with the 1-greedy strategy is that it also produces a very high standard deviation in the 10 executions performed, showing that a completely greedy strategy may produce very different results. The curve obtained by the Boltzmann strategy do not offer any improvements. However, the ϵ-greedy strategy seems to compute an accurate policy in the initials trials, and obtain the highest average gain at the end of the learning.

The random strategy and ϵ-greedy outperforms the other strategies in the test curve shown in Figure 3(b). This is due to the fact that both strategies, with the defined parameters, are less greedy than the other policies in the initial steps. Typically, higher exploration at the beginning results in more accurate policies.

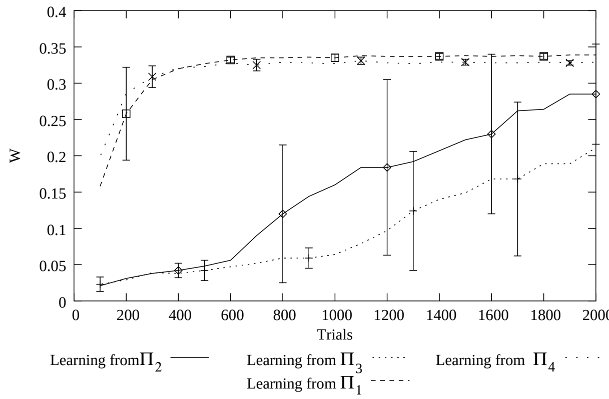
5.4 Reusing the Past Policy Following π-reuse

We want to learn to solve the task Ω, defined in Figure 2(e). To do this, we need to learn the action policy, Π_Ω that maximizes W_Ω, as defined in equation 1. In this case, we assume that a supervisor provides a similar task, say Ω_s, and the exploration strategy π-reuse is used to learn the new action policy.

Figure 4(a) shows the learning curves of different learning processes. In each of them, a different policy has been reused. As in the previous experiments, the parameters used are: $\gamma = 0.95$ and $\alpha = 0.05$ in the Q-Learning update function. $\psi = 1$ and $\upsilon = 0.95$, which empirically have demonstrated to be accurate. We distinguish three different cases. In the first one, the task previously learned is Ω_4 ($\Omega_s = \Omega_1$), the goal of which is into the same room as the goal of Ω. In the second case, $\Omega_s = \Omega_1$, so their goals are in different rooms. However, their optimal policies could be the same for all the domain except for the rooms where the respective goals are located. In the last two cases, $\Omega_s = \Omega_2$ and Ω_3 respectively, which are very different when compared to Ω.



(a) Learning Curve



(b) Test Curve

Figure 4: Learning and test evolution when following the exploration strategy π-reuse.

Figure 4(a) shows how, when biasing the exploration process for learning the task Ω with the policies Π_1 and Π_4, the obtained gain increases dramatically within the first few trials of the execution. For instance, when reusing Π_1, in only 100 iterations the average gain is higher than 0.15, and after 400 iterations the value stays around 0.2. When reusing Π_4, the gain is higher than 0.1 after only 200 trials, and after 500 trials it stays around 0.15. In both cases, the standard deviation is high in the initial trials, but it approaches 0 in subsequent trials. The behavior of the test curves is also very good in both cases, showing that in only 400 iterations, a gain higher than 0.3 is obtained with a very low deviation. These results demonstrate that reusing similar past policies produces a significant improvement over exploration strategies that learn from scratch.

However, when the learning is biased with a very different policy, as Π_2 and Π_3, the average gain shown in Figure 4(a) is below 0.05, so the learning process is even worse than when learning from scratch. Their test curves present a better behavior. In both cases there is an inflexion in the test curve, obtaining, at the end of the 2000 trials, a similar performance than unbiased strategies.

The inflexion is due to the learning of an initial path to the goal. However, in this case the standard deviation is very high, demonstrating that the inflexion may occur in very different moments of the learning process.

6 Similarity between Policies

Previous results show that reusing a past policy provides a bias in the exploration process which speeds up the learning. The improvement depends on whether the reused policy solves a task similar to the one we are currently learning. However, that is not the only benefit of policy reuse. One interesting observation about the results in Figure 4(a) is that the learning curves provide us with a very useful metric of similarity between policies. In that figure, the gain obtained for each of the past policies can be understood as: (i) an estimation of how similar the policy reused is to the current one; and (ii) an estimation of how useful the policy reused is in order to learn the new policy. Actually, the gain obtained by each one could be used to rank the similarity of the past policies with respect to the new one. In this case, the most similar is Π_4, followed by Π_1, Π_2 and Π_3.

Furthermore, the estimations above can be computed very fast, as Figure 5 demonstrates. The figure zooms in on the initial 100 trials of Figure 4(a). The figure shows that in only 25 trials, the gain of reusing the policy Π_4 significantly outperforms the gain of reusing the other policies. Thus, in a total of 100 trials (25 for each policy), the most similar policy, and therefore, the best policy to reuse, can be computed. These ideas are formalized next.

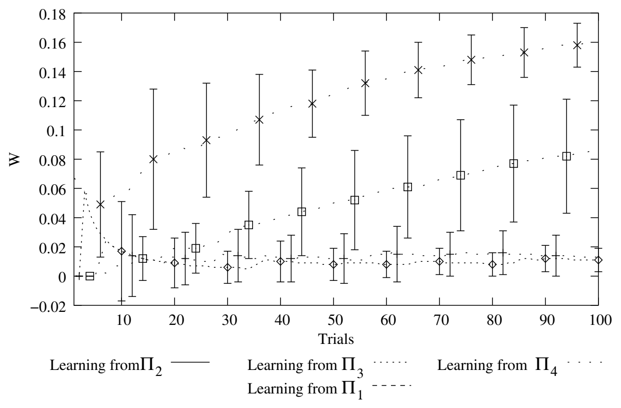


Figure 5: Computation of the Similarity Among Policies.

6.1 A Similarity Metric Between Policies

As introduced below, we call Π^*_Ω the optimal action policy for solving the task Ω. To be optimal means that $W_\Omega^{\Pi^*} \geq W_\Omega^{\Pi}$, for all policies Π in the space of all possible policies. Then, the following theorem can be derived.

Theorem: Given two different tasks, Ω_i and Ω_j, and their optimal policies, Π_{Ω_i} and Π_{Ω_j} respectively, then:

$$W_{\Omega_i}^{\Pi^*_{\Omega_i}} \geq W_{\Omega_i}^{\Pi^*_{\Omega_j}}, \forall \Omega_j \neq \Omega_i \tag{2}$$

It is easy to demonstrate this theorem, given that $\Pi^*_{\Omega_i}$ is the optimal policy for the task that we are involved in now, i.e, Ω_i. That ensures the maximum expected reward will be received, given that:
$V^{\Pi^*_{\Omega_i}}(s) \geq \mathcal{R}(s,a) + \sum_{s' \in \mathcal{S}} \delta(s,a,s') V^{\Pi^*_{\Omega_i}}(s')$, for $a = \arg_{a'} \max \Pi^*_{\Omega_j}(s,a')$.

The previous theorem ensures that $W_{\Omega_i}^{\Pi^*_{\Omega_i}} - W_{\Omega_i}^{\Pi^*_{\Omega_j}} \geq 0$. Therefore, we can define how useful the policy Π_i could be in learning the policy Π, using the distance metric shown in equation 3.

$$d_{\rightarrow}(\Pi_i, \Pi) = W_{\Omega}^{\Pi^*_{\Omega}} - W_{\Omega}^{\Pi^*_{\Omega_i}} \tag{3}$$

In this case the distance metric is not symmetric, so $d_{\rightarrow}(\Pi_i, \Pi_j)$ could be different from $d_{\rightarrow}(\Pi_j, \Pi_i)$. Then, the most useful policy to reuse is:

$$\arg_{\Omega_i} \min(W_{\Omega}^{\Pi^*_{\Omega}} - W_{\Omega}^{\Pi^*_{\Omega_i}}), i = 1, \ldots, n \tag{4}$$

However, $W_{\Omega}^{\Pi^*_{\Omega}}$ is independent of i, so the previous equation is equivalent to:

$$\arg_{\Omega_i} \max(W_{\Omega}^{\Pi^*_{\Omega_i}}), i = 1, \ldots, n \tag{5}$$

This equation is not possible to compute, given that $W_{\Omega}^{\Pi^*_{\Omega_i}}$ is unknown a priori. Furthermore, if we follow the policy $\Pi^*_{\Omega_i}$ greedily, probably the task Ω will never be solved. However, if instead of following $\Pi^*_{\Omega_i}$ greedily, we reuse it following the π-reuse exploration strategy, we can compute the gain of reusing $\Pi^*_{\Omega_i}$ to solve Ω. In this sense, all the past policies could be reused, computing their respective gains, until an accurate estimation is obtained. Then, past policies with a lower gain are discarded, and the one with a higher gain is used in the π-reuse exploration strategy. The next section describes a simple algorithm that applies these ideas.

6.2 An Algorithm for Policy Reuse

A basic algorithm for policy reuse from a set of policies requires the following two steps:

- Obtain the most similar policy, Π_s. To do this, it is necessary (i) to compute the gain obtained when following the π-reuse exploration policy with each of the past policies; and (ii) to choose the policy with a higher gain. We call K_s the number of trials used to learn the similar policy.

- Learn a new action policy, Π_Ω. Π_s is used in the π-reuse exploration strategy to learn a new action policy. We call K_r the number of trials used to learn the new policy.

The previous steps are formalized in Table 2 where we assume that π-reuse is a method that we can call with the parameters defined in Table 1.

Policy Reuse Algorithm

```
• Given:
    1. A set of $n$ tasks $\{\Omega_1, \ldots, \Omega_n\}$.
    2. Their respective optimal policies, $\{\Pi^*_{\Omega_1}, \ldots, \Pi^*_{\Omega_n}\}$ to solve them
    3. A new task, $\Omega$, that we want to solve
    4. A maximum number of steps per trial, $H$
    5. A maximum number of trials to execute, $K$
    6. The tuple of integer values, $< K_s, K_r >$, such as $K = K_s + K_r$
    7. The parameters $\psi$ and $\upsilon$ used in the exploration strategy $\pi$-reuse.
• for i=1 to n do
    1. Execute $\pi$-reuse($\Pi_i, K_s/n, H, \psi, \upsilon$).
    2. Obtain the associated gain, $W_i$.
• Set $\Pi_s = \arg_{\Pi_i} \max W_i$
• Learn $\Pi_\Omega$ by calling $\pi$-reuse($\Pi_s, K_r, H, \psi, \upsilon$)
```

Table 2: An Algorithm for Policy Reuse

6.3 Empirical Results

Figure 6 shows the learning curve obtained when the Policy Reuse algorithm is executed for two set of parameters, $K_s = 100$ and $K_r = 1900$, and $K_s = 400$ and $K_r = 1600$ respectively. This learning curve demonstrates that, even when the similar policy must be computed, policy reuse can be very useful to bias the exploration of a learning process, providing better performance than learning from scratch. The test curves correspond with the one shown in Figure 4(b) when reusing Π_4, but delayed k_s steps.

However, the success of policy reuse, when it is applied to speed up learning, depends on several factors. For instance, it requires the definition of the values of K_s and K_r that, at the same time, could depend of the domain, the task, and the number of past policies available. Furthermore, it does not make the most of the experience obtained; for instance, the experience used in the computation of the most similar policy could be used also to learn the new policy. Thus, more accurate algorithms should reduce the number of parameters used, and could outperform the results.

7 Conclusions and Further Research

In this report, we have described Policy Reuse as an exploration bias that balances the exploration of random actions, the exploitation of the ongoing learned policy, and the exploration toward of a

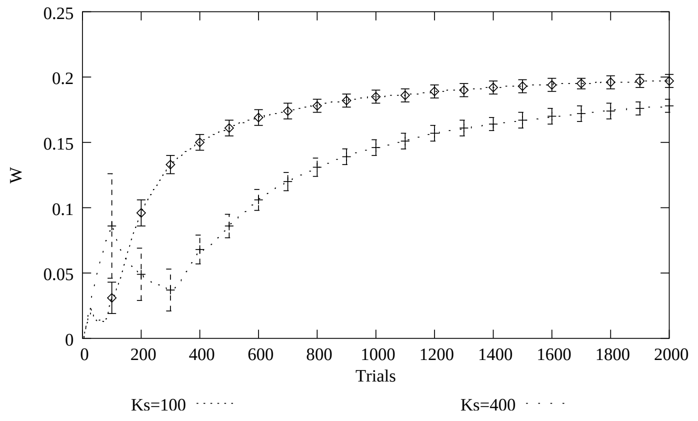


Figure 6: Learning Curve when following the Policy Reuse algorithm.

past policy. We have instantiated the concept of Policy Reuse by defining a new exploration strategy, π-reuse. This new strategy successfully achieves the previous balance, and has demonstrated that it can improve the learning performance obtained when learning from scratch with different strategies.

Furthermore, we have demonstrated that Policy Reuse provides a similarity metric between policies. Such a metric allows to discriminate, from a set of past policies, which is the most similar one to the policy we currently are learning.

Policy Reuse and the concept of similarity introduced in this work open a wide range of challenging research lines, including across domain or agent learning and the ability to scale RL in complexity considerably.

References

[1] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In IEEE Computer Society Press, editor, *36th Annual Symposium on Foundations of Computer Science*, pages 322–331, 1995.

[2] Michael Bowling and Manuela Veloso. Bounding the suboptimality of reusing subproblems. In *Proceedings of IJCAI-99*, 1999.

[3] James Bruce and Manuela Veloso. Real-time randomized path planning for robot navigation. In *Proceedings of IROS-2002*, Switzerland, October 2002. An earlier version of this paper appears in the Proceedings of the RoboCup-2002 Symposium.

[4] James Carroll and Todd Peterson. Fixed vs. dynamic sub-transfer in reinforcement learning. In *Proceedings of the International Conference on Machine Learning and Applications*, 2002.

[5] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[6] Fernando Fernández and Daniel Borrajo. On determinism handling while learning reduced state space representations. In *Proceedings of the European Conference on Artificial Intelligence (ECAI 2002)*, Lyon (France), July 2002.

[7] M. L. Puterman. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY., 1994.

[8] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin American Mathematical Society*, 55:527–535, 1952.

[9] Richard S. Sutton, Doina Precup, and Satinder Singh. Intra-option learning about temporally abstract actions. In *Proceedings of the Internacional Conference on Machine Learning (ICML'98)*, 1998.

[10] Sebastian Thrun. Efficient exploration in reinforcement learning. Technical Report C,I-CS-92-102, Carnegie Mellon University, January 1992.

[11] Sebastian Thrun and Tom Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15:25–46, 1995.

[12] Sebastian Thrun and A. Schwartz. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7*. MIT Press., 1995.

[13] William T. B. Uther. *Tree Based Hierarchical Reinforcement Learning*. PhD thesis, Carnegie Mellon University, August 2002.

[14] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.